

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №1.2

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Исследование возможностей Git для работы с локальными
репозиториями»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

Кочкаров Умар Ахматович

Ставрополь 2022

Выполнение работы:

1. Создан общедоступный репозиторий на GitHub.

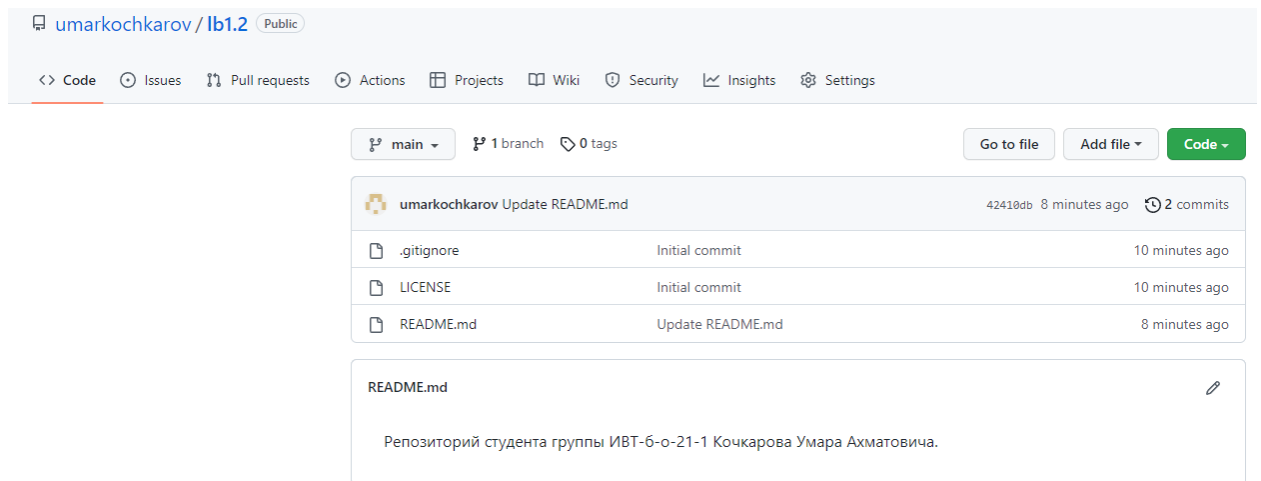


Рисунок 1. Созданный репозиторий

2. Добавлена информация в README, выполнен коммит и пуш.

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/lb1.2/lb1.2
(main)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/lb1.2/lb1.2
(main)
$ git add .

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/lb1.2/lb1.2
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/lb1.2/lb1.2
(main)
$
```

Рисунок 2. Коммит README

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git commit -m "modified README"
[main 5e6363e] modified README
1 file changed, 1 insertion(+)

```

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 414 bytes | 414.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/umarkochkarov/1b1.2.git
8956d56..5e6363e  main -> main

```

Рисунок 2,3,4. Коммит и пуш README

main
1 branch
0 tags
Go to file
Add file
Code

umarkochkarov modified README	5e6363e 2 minutes ago	4 commits
.gitignore	Update .gitignore	2 days ago
LICENSE	Initial commit	2 days ago
README.md	modified README	2 minutes ago

README.md

Репозиторий студента группы ИВТ-6-о-21-1 Кочкарова Умара Ахматовича. Дисциплина: Основы кроссплатформенного программирования

Рисунок 5. Изменения на удаленном сервере

3. Написана небольшая программа в репозитории.

```

1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          cout << "Hello world!" << endl;
8          return 0;
9      }

```

Рисунок 6. Код

.git	06.09.2022 19:56	Папка с файлами	
prog	06.09.2022 20:08	Папка с файлами	
.gitignore	05.09.2022 0:04	Текстовый докум...	8 КБ
LICENSE	05.09.2022 0:04	Файл	2 КБ
README	05.09.2022 10:27	Исходный файл ...	1 КБ

Рисунок 7. Папка с программой

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   prog/prog.cpp

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git add .

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git commit -m "added prog"
[main 3519c0d] added prog
1 file changed, 9 insertions(+)
create mode 100644 prog/prog.cpp

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 400 bytes | 400.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/umarkochkarov/1b1.2.git
5e6363e..3519c0d main -> main

```

Рисунок 8. Коммит и пуш программы

umarkochkarov added prog		3519c0d 1 minute ago	🕒 5 commits
📁 prog	added prog		1 minute ago
📄 .gitignore	Update .gitignore		2 days ago
📄 LICENSE	Initial commit		2 days ago
📄 README.md	modified README		17 minutes ago

Рисунок 9. Изменения на сервере

4. Просмотрен журнал хранилища при помощи git log:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git log --graph --pretty=oneline --abbrev-commit
* 03ba2c7 (HEAD -> main, origin/main, origin/HEAD) programm My name is Umar
* 3519c0d added prog
* 5e6363e modified README
* 8956d56 Update .gitignore
* 42410db Update README.md
* 125e6f5 Initial commit

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ .....
```

Рисунок 10. История коммитов

5. При помощи команды git show <> просмотрен последний коммит:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git show HEAD
commit 03ba2c7a326e91830ed6f61c7485689b513b4447 (HEAD -> main, origin/main, origin/HEAD)
Author: umarkochkarov <kochkarov-u@mail.ru>
Date: Sat Sep 10 18:35:19 2022 +0300

    programm My name is Umar

diff --git a/prog/prog.cpp b/prog/prog.cpp
index b4392ec..e1ea40c 100644
--- a/prog/prog.cpp
+++ b/prog/prog.cpp
@@ -5,5 +5,6 @@ using namespace std;
 int main()
 {
     cout << "Hello world!" << endl;
+    cout << "My name is Umar" << endl;
     return 0;
 }
```

Рисунок 11. Последний коммит

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git show HEAD~1
commit 3519c0dc5ab32fdd8c785c4db916e27a1b0e4961
Author: umarkochkarov <kochkarov-u@mail.ru>
Date: Tue Sep 6 20:11:41 2022 +0300

    added prog

diff --git a/prog/prog.cpp b/prog/prog.cpp
new file mode 100644
index 0000000..b4392ec
--- /dev/null
+++ b/prog/prog.cpp
@@ -0,0 +1,9 @@
+#include <iostream>
+
+using namespace std;
+
+int main()
+{
+    cout << "Hello world!" << endl;
+    return 0;
+}

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$

```

Рисунок 12. Предпоследний коммит

6. Просмотр коммита с определенным хэшем:

```

$ git show 03ba2c7a326e91830ed6f61c7485689b513b4447
commit 03ba2c7a326e91830ed6f61c7485689b513b4447 (HEAD -> main, origin/main, orig
in/HEAD)
Author: umarkochkarov <kochkarov-u@mail.ru>
Date: Sat Sep 10 18:35:19 2022 +0300

    programm My name is Umar

diff --git a/prog/prog.cpp b/prog/prog.cpp
index b4392ec..e1ea40c 100644
--- a/prog/prog.cpp
+++ b/prog/prog.cpp
@@ -5,5 +5,6 @@ using namespace std;
 int main()
 {
     cout << "Hello world!" << endl;
+    cout << "My name is Umar" << endl;
     return 0;
 }

```

Рисунок 13. Просмотр коммита с определенным хэшем

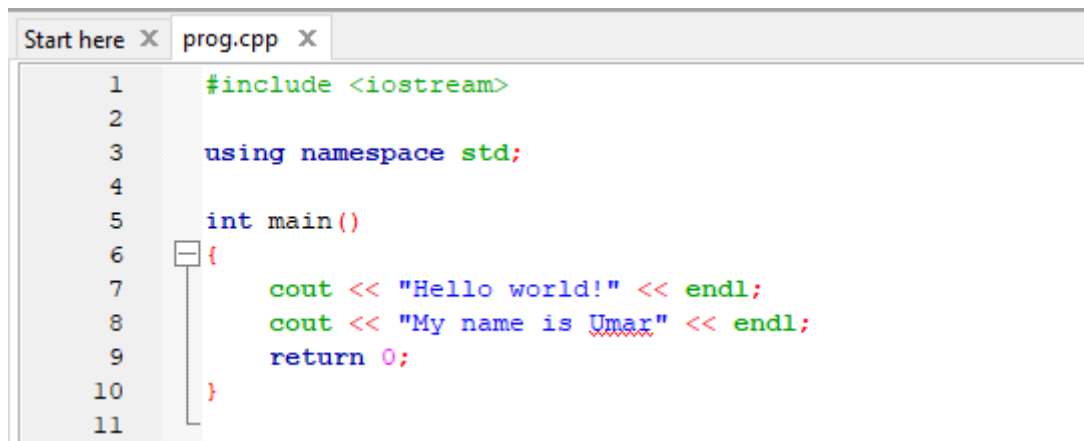
7. Удалил код файла prog и сделал коммит.
8. С помощью команды `git reset` откатил состояние хранилища к предыдущей версии:

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Мои файлы/Универ/Основы крос/1b1.2/1b1.2
(main)
$ git reset --hard HEAD~1
HEAD is now at 03ba2c7 programm My name is Umar

```

Рисунок 14. Команда `git reset`



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      cout << "My name is Umar" << endl;
9      return 0;
10 }
11
```

Рисунок 15. Восстановленный код

Ответы на контрольные вопросы:

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список коммитов созданных в данной репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми.

Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `–stat`.

Вторая опция (одна из самых полезных аргументов) является `-p` или `-- patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log -p -2`).

Третья действительно полезная опция это `--pretty`. Она меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удоб-

ным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно.

Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git.

Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log <n>`, где `n` число записей.

Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели:

```
git log --since=2.weeks
```

Это команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`.

Также вы можете фильтровать список коммитов по заданным параметрам. Опция `--author` дает возможность фильтровать по автору коммита, а опция `--grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита. Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit --amend`

Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту.

Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`.

```
git commit -m 'initial commit' git add forgotten_file
```

```
git commit --amend
```

Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам:

Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>` для исключения из индекса.

5. Как отменить изменения в файле?

С помощью команды `git checkout -- <file>`.

6. Что такое удаленный репозиторий Git?

Удалённый репозиторий это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, необходимо запустить команду `git remote`.

Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду `git remote add <shortname> <url>`.

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch <Название репозитория>`. Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и немодифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы.

Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (fetch) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете.

Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push <remote-name> <branch-name>`.

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show <remote>`.

11. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии кода/написанной программы. Они удобны чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно пометать важные моменты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`.

А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`.

С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`.

Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin <tagname>`. Для отправки всех тегов можно использовать команду `git push origin tags`.

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее лёгкий тег можно следующим образом: `git tag -d v1.4-lw`

Для удаления тега из внешнего репозитория используется команда `git push origin --delete <tagname>`.

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

13. Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага?

`Git fetch --prune` команда получения всех изменений с репозитория GitHub.

В команде `git push --prune` удаляет удалённые ветки, у которых нет локального аналога.